HISTORY REPEATS ITSELF: SENSIBLE AND NONSENSQL ASPECTS OF THE NOSQL HOOPLA

A RETROSPECTIVE CRITIQUE

04/11/2026

Ahmed El Shatshat

UNIVERSITY OF WATERLOO | FACULTY OF MATHEMATICS

MOTIVATION

Why the shift towards NoSQL?

WEB 2.0

- Large corporations were pushing for new database technology to support a global load of internet traffic
- Systems and services needed to be online and appear responsive at all times
  - Ground truth and consistency were thus seen as less important
- Throughput was a priority
  - Some internet data was explicitly ephemeral, and so could be stored in RAM
- CAP Theorem
  - Proponents argued that, by sacrificing Consistency, one could maximize Availability through Partitioning

HARDWARE AND DESIGN ADVANCEMENTS

- Data storage devices had gotten much cheaper to produce and more compact with storage
- Most relational database management systems (RDBMSs) like DB2 were designed for monolithic Symmetric Multi-Processing systems
  - With increased data and the advent of distributed computing, multi-system clustered database systems were being considered

LAZINESS

- RDBs required schemas and the creation of diagrams and models to accurately map out and analyze the database structure
  - NoSQL on the other hand just lets you throw data on the pile
- Unwillingness to put effort into developing or improving RDBMSs that can scale with the increased load
  - Though, as we'll see later, the idea that RDBMSs can't scale isn't necessarily true
- Corporations had large workforces they could mobilize to create in-house solutions to whatever problems arose

CLEARLY…

- There are some clear benefits to developing an alternative to classical RDBMSs
- However, in practice, the way in which these alternatives were developed was myopic and almost purposefully ignorant

A CRITICAL RETROSPECT

What happened in practice?

NOT REALLY REVOLUTIONARY

- NoSQL databases are typically built as either key-value stores (aka a dictionary), or as a column store (aka a 2D array)
  - Even contemporaneous writers were noting that the fanfare and hype were strange considering that these concepts aren't novel
- Much of the rhetoric from NoSQL proponents around RDBMSs act as though the technology:
  - Is monolithic in design ethos
  - Hasn't changed and improved since the 1970s
  - Doesn't already include some of the features they argue are the motivation behind leaving RDBs behind

PURPOSEFUL IGNORANCE

- Fast throughput doesn't justify inconsistency in writing data
  - Many of these systems had silent failures with writes
  - If throughput is all you care about, just write to /dev/null
- Burden of manipulating and solving data-related issues shifted from database admins to developers
  - Lack of typing (e.g. storing everything as JSON strings)
  - No query optimization leveraging relational database design
- No consistency in design or in syntax between NoSQL DBMSs
  - This can also apply self-referentially, as these systems evolve and mutate

FURTHER SLOPPINESS

- Access control
- Systems that use buffers for writing data can be exploited (MongoDB scandal)
- Benefits from hash indexing only apply with single-entry indices
  - This forces a very inefficient hashing structure
- Many alleged benefits of NoSQL systems take naïve approaches to problems and handwave over important design considerations and limitations
  - E.g. treating CAP Theorem as a simple pick-two

SO…

- There are many critical issues with how NoSQL systems were being developed
  - Both at a conceptual and at a material level
- So how did we get to a paradigm dominated by such systems?

WHY NOSQL CAME TO DOMINATE

According to me

CAPITALISM

- The push behind NoSQL was headed by business leaders and corporate mouthpieces (Google, Facebook, Last.fm)
  - The goal was short-term maximization of profit, rather than proper engineering
- The growing cheapness of data enabled actors with large budgets to leverage it into in-house technology
- Similarly, there was the growing idea that, in a reversal of the past, that computers were cheap and workers were expensive
  - You can avoid hiring people to administrate a proper database through brute force
  - Back in those days, the government was bankrolling these endeavors

PROPAGANDA

- There are actual use cases where a non-relational database would be preferable
  - Many of these use cases were still addressed using a naïve approach
- The apparent ease of use of NoSQL systems hide insidious drawbacks that were not commonly discussed
- DBs of old were either wholly ignored (e.g. Lotus Notes) or scrutinized with twice the vigor compared to proposed NoSQL solutions
  - This is coupled with the misconception, or disinformation, that RDBs haven't changed since the 70s, or that non-relational solutions also already existed
- Theory was pushed as equal to practice
  - Ironically, MySQL ended up performing at equivalent or better at partitioning than competing NoSQL systems

STONEBRAKER ET AL.

"'Self-everything' (self-healing, self-maintaining, self-tuning, etc.) [database] systems are the only answer. However, all RDBMSs have a vast array of complex tuning knobs, which are legacy features from a bygone era."

CAN YOU BELIEVE STONEBRAKER WAS INVOLVED IN MANY CORPORATE NOSQL PROJECTS?

CONSPIRACY THEORY

- Considering websites like Facebook, Twitter, and Google have been revealed to be no more than complex data-harvesting schemes, perhaps there was a more insidious push behind schema-less design
- For example, if I illegally sell data, and I'm investigated, a schema will be the smoking gun leading back to my company
- Furthermore, why work on representing data in a clever way when all you want to do is maximize the amount of data you're harvesting
  - Cleaning up the data is the job of the guy you're selling it to

WHAT IS TO BE DONE?

WHAT IS TO BE DONE

- NoSQL-based systems need to be more honest about their use cases and drawbacks
  - Rather than as an "alternative" to relational databases, they need to be seen for what they are: sloppier solutions to problems for cases when the benefits of acting sloppy outweigh the drawbacks
- There should be a standardization across NoSQL systems
- A major problem currently is that databases are thought of as a binary (SQL vs NoSQL) (Relational vs Not), with some major subcategories (key-value, column-store, graph etc.)
  - However, even within subcategories there can be a lot of variance in design and scope amongst products
  - An academic approach assessing clear heuristics and design paradigms would help cut through the confusion

BIBLIOGRAPHY

- C. Mohan, “History repeats itself,” International Conference on Extending Database Technology (EDBT), pp. 11–16, Mar. 2013, doi: 10.1145/2452376.2452378.
- M. Stonebraker, S. Madden, D. J. Abadi, S. Harizopoulos, N. Hachem, and P. Helland, “The end of an architectural era: it’s time for a complete rewrite,” in ACM eBooks, 2018, pp. 463–489. doi: 10.1145/3226595.3226637.
- C. Strauch, “NoSQL databases,” Hochschule der Medien, Stuttgart, Lecture, 2010. [Online]. Available: https://www.christof-strauch.de/nosqldbs.pdf
- R. Cattell, “Scalable SQL and NoSQL data stores,” ACM SIGMOD Record, vol. 39, no. 4, pp. 12–27, May 2011, doi: 10.1145/1978915.1978919.
- Akamai Security Intelligence Group, “CVE-2025-14847: All You Need to Know About MongoBleed,” Akamai, Dec. 25, 2025. https://www.akamai.com/blog/security-research/cve-2025-14847-all-you-need-to-know-about-mongobleed (accessed Nov. 03, 2026).



Thank you!